PERL scripting

Perl is a scripting language
It is compiled before each run
To tell UNIX that following is perl script write #!/usr/bin/perl at the very first line of perl script

** Comments in Perl
****
Single line comment is written starting with #
Commands starts with first non space character and ends with ;
subroutine is written like sub subname{command;}
call subroutine by &subname;
pass arugments like &subname(@data);
inside subroutine receive arguments like
sub subname{
@x = @data;
commands;
}
sub subname{
($var1,$var2,$var3) = @data;
commands;
}
sub subname{
lcoal($var1,$var2,$var3) = @data;
commands;
}
** Quotes in Perl
****
single quote is written as ''
double quote is written as ""
quote for execution is ``
quote which contains / is written as qq!<quote with />!

** Variables in Perl
****
no data type for variable, perl switches data type as required
$variableName = value; is used to define a variable with value ex. $x =10; $y="XYZ";
@variable for array variable, ex. @x = (10, 20, 30);
individual element of an array accessed as $x[2]
%variable for associative array variable, ex. %x = ('name','XYZ', 'age',30, 'key','value'); or %x =
(name=>'XYZ', age=>30, key=>value);
individual element of an associative array accessed as $x{"key"} returns value

** assignments in Perl
****
$var = value; # normal assignment
$var .= stringValue; # string appending
$var += value; # add value to previous value of $var, also called as short hand assignment

PERL scripting

********** comparision in Perl
****
to compare numbers == != > < <= >=
to compare strings eq (for equal?) ne (for not equal?)

********** logical operators in Perl
****
&& (AND), || (OR), ! (NOT)
instead of writing
if !$var
one can write
unless $var

********** if and unless in Perl
****
```
# for if-else ladder
if(condition){
   command;
}elsif(condition){
   command;
}else{
   command;
}

# for single line command with condition
command if condition;

# opposite of if
unless(condition){
   command;
}else{
   command;
}

# for single line command with unless condition
command unless condition;
```

********** looping in Perl
****
```
while(condition){
   command;
}

# Go prematurely to the next iteration
while(condition){
   command;
   next if condition;
```

```
   command;
}

# Prematureley abort the loop with last
while(condition){
   command;
   last if condition;
}

# until (just the opposite of while)
until(condition){
   command;
}

until(condition){
   command;
   next if condition;
   command;
}

until(condition){
   command;
   last if condition;
}

# Iterate over @data and have each value in $_
for(@data){
   print $_,"\n";
}

# Get each value into $info iteratively
for $info (@data){
   print $info,"\n";
}

# Iterate over a range of numbers
for $num (1..100){
   next if $num % 2;
   print $num,"\n";
}

# Eternal loop with (;;)
for (;;){
   $num++;
   last if $num > 100;
}
```

PERL scripting

```
****************************************** substitututions in Perl
**************************************************
$var =~s/patten/replace;
($newVar=$oldVar) =~s/pattern/replace;

****************************************** print in Perl
**************************************************
print $var, "\n";
printf("%s %d",$str, $var);

# output in a file
open(OUT, "<fileName>") || die "error message";
print OUT $var;
close OUT;

# print a file if it is not empty
open(IN,"<file") || die "Cannot open file\n";
if(eof(IN)){
   print "File is empty\n";
}else{
   while(<IN>){
      print;
   }
}
close IN;

****************************************** read in Perl
**************************************************
chomp($var=<STDIN>); # removes new line character from last and puts in $var

# to read from file
oepn(IN, "<fileName>") || die "error message";
while(<IN>){
commands;
}
close IN;

# to read a file into array
open(IN, "<fileName>") || die "error message";
@arr = <IN>;
close IN;

# Process Files mentioned on the Commandline
while(<>){
   $file = $ARGV;
   print $file,"\t",$_;
   open(IN,"<$file") or warn "Cannot open $file\n";
   ....commands for this file....
```

```
   close(IN);
}

****************************************** built in functions in Perl
************************************************
Get all upper case with: $name = uc($name);
Get only first letter uppercase:   $name = ucfirst($name);
Get all lowercase:          $name = lc($name);
Get only first letter lowercase:   $name = lcfirst($name);
Get the length of a string:          $size = length($string);
Extract 5-th to 10-th characters from a string:     $part = substr($whole,4,5);
Remove line ending:      chomp($var);
Remove last character:  chop($var);
Crypt a string:    $code = crypt($word,$salt);
Execute a string as perl code:     eval $var;
Show position of substring in string:       $pos = index($string,$substring);
Show position of last substring in string: $pos = rindex($string,$substring);
Quote all metacharachters:        $quote = quotemeta($string);

Array Functions
Get expressions for which a command returned true:      @found = grep(/[Jj]ohn/,@users);
Applay a command to each element of an array: @new = map(lc($_),@start);
Put all array elements into a single string:          $string = join(' ',@arr);
Split a string and make an array out of it:           @data = split(/&/,$ENV{'QUERY_STRING'};
Sort an array:    sort(@salery);
Reverse an array:          reverse(@salery);
Get the keys of a hash(associative array):          keys(%hash);
Get the values of a hash:          values(%hash);
Get key and value of a hash iteratively:   each(%hash);
Delete an array:@arr = ();
Delete an element of a hash:     delete $hash{$key};
Check for a hash key:     if(exists $hash{$key}){;}
Check wether a hash has elements:        scalar %hash;
Cut of last element of an array and return it:       $last = pop(@IQ_list);
Cut of first element of an array and return it:      $first = shift(@topguy);
Append an array element at the end:      push(@waiting,$name);
Prepend an array element to the front:  unshift(@nowait,$name);
Remove first 2 chars an replace them with $var: splice(@arr,0,2,$var);
Get the number of elements of an array:          scalar @arr;
Get the last index of an array:    $lastindex = $#arr;

File Functions
Open a file for input:     open(IN,"</path/file") || die "Cannot open file\n";
Open a file for output:   open(OUT,">/path/file") || die "Cannot open file\n";
Open for appending:     open(OUT,">>$file") || &myerr("Couldn't open $file");
Close a file:       close OUT;
Set permissions:          chmod 0755, $file;
Delete a file:      unlink $file;
```

PERL scripting

Rename a file: rename $file, $newname;
Make a hard link: link $existing_file, $link_name;
Make a symbolic link: symlink $existing_file, $link_name;
Make a directory: mkdir $dirname, 0755;
Delete a directory: rmdir $dirname;
Reduce a file's size: truncate $file, $size;
Change owner- and group-ID: chown $uid, $gid;
Find the real file of a symlink: $file = readlink $linkfile;
Get all the file infos: @stat = stat $file;

Conversions Functions
Number to character: chr $num;
Charachter to number: ord($char);
Hex to decimal: hex(0x4F);
Octal to decimal: oct(0700);
Get localtime from time: localtime(time);
Get greenwich meantime: gmtime(time);
Pack variables into string: $string = pack("C4",split(/\./,$IP));
Unpack the above string: @arr = unpack("C4",$string);

Addition: +
Subtraction: -
Multiplication: *
Division: /
Rise to the power of: **
Rise e to the pwoer of: exp()
Modulus: %
Square root: sqrt()
Absolut value: abs()
Tangens: atan2()
Sinus: sin()
Cosine: cos()
Random number: rand()

Command line Switches
Show the version number of perl: perl -v;
Check a new program without runing it: perl -wc <file>;
Have an editing command on the command line: perl -e 'command';
Automatically print while precessing lines: perl -pe 'command' <file>;
Remove line endings and add them again: perl -lpe 'command' <file>;
Edit a file in place: perl -i -pe 'command' <file>;
Autosplit the lines while editing:perl -a -e 'print if $F[3] =~ /ETH/;' <file>;
Have an input loop without printing: perl -ne 'command' <file>;